Claims listing:

1. (currently amended) A method for ~~transforming test cases that are converted to an abstract representation and storing abstract representation of~~ generating platform-specific test ~~cases~~ scripts for an application ~~into a database system~~, the method comprising:

~~storing test cases in abstract representations to generate test cases in any target environment script format to provide interoperability between automation tools and cross-environment portability of test cases;~~

importing at least one test case~~s~~ written in at least one ~~or more~~ scripting language~~s~~;

~~using semantic analysis to~~ converting the at least one test case~~s~~ to an abstract representation using semantic analysis, wherein the abstract representation comprises at least three components, and wherein the at least three components include at least one ~~that includes~~ application state, at least one external interaction sequence~~s~~, and input data ~~without changing or deleting an original test case;~~, and wherein the at least one application state ~~is~~ comprises at least one of: (a) a set of application objects associated with a set of attributes and their values, or (b) ~~represents~~ a runtime snapshot of ~~an~~ the application while under test which defines a context of external interaction,

~~using environment mappings to provide platform independence of test case and the abstract representation providing a platform independent representation of test case, the test cases being recombined and modified using external rules to combine and modify components of the abstract representation of test cases into new scripts;~~

storing the abstract representation ~~of test cases~~ into a database system; and

generating at least one platform-specific test script~~s~~ ~~for multiple test execution environments~~ based on the abstract representation using an environmental mapping.

2. (currently amended) The method of claim 1, wherein ~~an~~ the at least one application state ~~represents~~ comprises a representation of a runtime snapshot of the application while under test ~~which defines the context of external interaction~~.

3. (currently amended) The method of claim 2, wherein the at least one application state ~~includes~~ comprises a set of application objects, ~~its~~ attributes of the application objects, and values of the attributes ~~values~~.

4. (currently amended) The method of claim 2, wherein the at least one application state is a plurality of application states, and the plurality of application states ~~corresponding to a test case~~ are arranged in a hierarchical manner.

5. (original) The method of claim 2, wherein the database system is a relational database management system.

6. (previously presented) The method of claim 2, wherein the database system is an extensible markup language (XML) database management system.

7. (currently amended) The method of claim 2, wherein the at least one scripting language~~s can be~~ is at least one of a typed programming language or an untyped programming language~~s~~ used for at least one of recording or authoring test cases.

8. (currently amended) The method of claim 2, wherein the at least one external interaction sequences ~~represent~~ comprises a representation of events invoked by external agents on the set of application objects.

9. (currently amended) The method of claim 8, wherein at least one external agent~~s are~~ is at least one of a human agent~~s~~ or ~~other~~ a software agent~~s~~.

10. (currently amended) The method of claim 8, wherein the at least one interaction ~~sequencing includes~~ sequence comprises at least one flow control structure~~s~~ for capturing at least one of a sequential interaction, a concurrent interaction, a looping interaction, or a ~~and~~ conditional interaction~~s~~.

11. (currently amended) The method of claim 2, further comprising:

implementing ~~a~~ at least one syntax analyzer for ~~incoming scripts~~ the at least one test case.

12. (currently amended) The method of claim 11, wherein ~~the~~ a distinct syntax analyzer is implemented ~~one~~ for each scripting language used in each test case.

13. (currently amended) The method of claim 12, wherein the at least one syntax analyzer ~~utilizes~~ comprises rules of syntax analysis that are ~~specified~~ provided in Extended Backus-Naur Form ~~(EBNF)~~.

14. (currently amended) The method of claim 12, further comprising generating at least one ~~wherein the syntax analyzer generates a~~ parse tree in the form of an Abstract Syntax Tree ~~(AST)~~ using the at least one syntax analyzer.

15. (currently amended) The method of claim 2, ~~further comprising:~~

~~implementing a~~ wherein the semantic analysis ~~that~~ is used to converts an ~~abstract syntax tree~~ Abstract Syntax Tree to ~~an~~ the abstract ~~test case~~ representation based on an Application Object Model ~~(AOM)~~.

16. (currently amended) The method of claim 15, wherein the semantic analysis is used to decomposes the ~~test cases represented as an~~ Abstract Syntax Tree into the at least one application state, the at least one external interaction sequences and the input data.

17. (currently amended) The method of claim 15, wherein ~~an application object model is~~ the Application Object Model comprises a metadata representation ~~for modeling~~ of the application ~~under test~~.

18. (currently amended) The method of claim 17, wherein the metadata representation ~~includes~~ comprises object type definitions for application objects.

19. (currently amended) The method of claim 17, wherein the metadata representation ~~includes~~ comprises attribute definitions for each type of application object ~~type~~.

20. (currently amended) The method of claim 17, wherein the metadata representation ~~includes~~ comprises definitions of a plurality of methods and events that are supported by each type of application object ~~type~~.

21. (currently amended) The method of claim 17, wherein the metadata representation ~~includes~~ comprises definitions of a plurality of effects of events on an application state.

22. (currently amended) The method of claim 18, wherein ~~application~~ the object type definitions ~~include additional categorization of each application object types into~~ comprise hierarchical object types, container object types, and simple object types.

23. (currently amended) The method of claim 22, wherein ~~the~~ each hierarchical object type~~s are~~ is associated with ~~an~~ a distinct application state ~~of its own~~; and wherein each container object type comprises an application object type~~s that can~~ which is configured to contain instances of other application objects ~~are termed as container types~~.

24. (currently amended) The method of claim 23, wherein the state associated with a hierarchical application object type is at least one of a modal application state or a nonmodal application state.

25. (currently amended) The method of claim 24, wherein ~~a~~ the modal application state is configured to restrict~~s~~ possible interactions ~~to~~ with application object instances available within ~~the~~ a current application state.

26. (currently amended) The method of claim 22, wherein the effects of events on ~~an~~ the at least one application state capture at least one ~~or more~~ consequence~~s~~ of the event~~s~~ to the application state.

27. (currently amended) The method of claim 26, wherein ~~a~~ the at least one consequence of an event ~~is selected from,~~ comprises at least one of creation of a new object instance of a given type, deletion of an object instance of a given type, modification of attributes of an existing object instance, or ~~and~~ selection of an instance of an object type.

28. (currently amended) The method of claim 27, wherein the creation of a new object instance ~~of~~ for ~~an object of type that is~~ a hierarchical object ~~results in~~ comprises creation of a new application state.

29. (currently amended) The method of claim 27, wherein the selection of an ~~object~~ instance of an object type that is hierarchical ~~results in~~ comprises selection of the application state associated with ~~that object~~ the instance.

30. (currently amended) The method of claim 2, further comprising:

enriching the abstract representation ~~of test cases with~~ based on information from an application metadata repository.

31. (currently amended) The method of claim 30, wherein ~~the enrichment of abstraction~~ enriching the abstract representation ~~of test cases involves~~ comprises extracting values for ~~those~~ attributes of application objects associated with the at least one test case~~s~~ that are missing ~~in the incoming test scripts~~.

32. (currently amended) The method of claim 30, wherein enriching the ~~abstraction~~ abstract representation ~~of test cases includes~~ comprises decoupling ~~of~~ the at least one test case~~s~~ from ~~their~~ at least one of an associated recording environment or an associated authoring environment~~s~~.

33. (currently amended) The method of claim 30, wherein enriching the ~~abstraction~~ abstract representation ~~of test cases allows usage of~~ provides attributes that are stable within an application metadata representation.

34. (currently amended) The method of claim 33, wherein ~~using~~ at least one application object is identified by an identification field ~~for a given object~~ within the application metadata repository ~~improves the reusability of a test case instead of a label used to represent the same object within a user interface which can change based on the locale of the application~~.

35. (currently amended) The method of claim 33 wherein ~~using an~~ the identification field ~~allows to overcome the problem of different test execution environments using different attributes to identify the same application object~~ provides platform independence of the abstract representation.

36. (currently amended) The method of claim 35, wherein enriching the ~~abstraction~~ abstract representation ~~of test cases enables~~ provides a representation of test cases that ~~are~~ is independent of any particular test execution environment ~~independent~~.

37. (currently amended) The method of claim 2, further comprising:

separating application object attributes and input data from external interaction sequencing to provide~~s~~ automatic parameterization.

38. (currently amended) A system for ~~transforming~~ providing an abstract representation of at least one test case~~s to an abstract representation that are stored in a database~~, comprising:

a processor~~;~~ ~~for importing test cases written in one or more scripting languages~~;

a memory arrangement coupled to the processor, the memory arrangement configured to store the at least one test case;

~~logic~~ a first set of instructions which, when executed by the processor, configures the processor to use ~~that includes~~ semantic analysis ~~for~~ to convert~~ing~~ the at least one test cases to an abstract representation, wherein the abstract representation comprises at least three components, and the at least three components include ~~that includes~~ at least one application state, at least one external interaction sequence~~s~~, and input data~~;~~ ~~and a database that stores test cases in abstract representations to generate test cases in any target environment script format to provide interoperability between automation tools and cross environment portability of test cases~~, and wherein the application state ~~being~~ comprises at least one of (a) a set of application objects associated with a set of attributes and their values, or (b) ~~represents~~ a runtime snapshot of ~~an~~ the application while under test which defines a context of external interaction,

~~the logic using~~ a second set of instructions which, when executed by the processor, configures the processor to apply environment mappings ~~providing platform independence of test cases and~~ to the abstract representation to provide at least one test script~~s are generated for multiple~~ configured to be executed in a particular one of a plurality of test execution environments without changing or deleting an original test case, the test cases being recombined and modified using external rules to combine and modify components of the abstract representation of test cases into new scripts.

39. (currently amended) The system of claim 38, ~~further comprising:~~ wherein the at least one test case is converted using a syntax analyzer ~~for incoming scripts~~.

40. (currently amended) The system of claim 38, ~~further comprising:~~

~~logic for implementing a~~ wherein the semantic analysis ~~that~~ is configured to converts ~~the~~ an abstract syntax tree to ~~an~~ the abstract ~~test case~~ representation based on an Application Object Model ~~(AOM)~~.

41. (canceled)

42. (currently amended) The system of claim 38, further comprising:

~~logic for enriching~~ a third set of instructions which, when executed by the processor, configures the processor to enrich the abstraction representation ~~of test cases~~ to ~~enable~~ provide a representation of the at least one test case~~s~~ that ~~are~~ is independent of any particular test execution environment ~~independent~~.

43. (currently amended) A computer system for storing an abstract representation~~s~~ of at least one test case~~s~~ in a database, comprising:

a processor;

a memory arrangement coupled to the processor, wherein the memory arrangement is configured to store:

(i) the at least one test case, and

(ii) a first set of instructions which, when executed by the processor, configures the processor to convert~~ing~~ the at least one test case~~s~~ to an abstract representation, wherein the abstract representation comprises at least three components, and wherein the at least three components include ~~that includes~~ at least one application state, at least one external interaction sequence~~s~~, and input data, and wherein the at least one application state ~~being~~ comprises at least one of (a) a set of application objects associated with a set of attributes and their values, or (b) ~~represents~~ a runtime snapshot of an application while under test which defines a context of external interaction, ~~the test cases being recombined and modified using external rules to combine and modify components of the abstract representation of test cases into new scripts without changing or deleting an original test case~~, and

a database ~~that~~ configured to stores ~~s~~ the abstract representation ~~of test cases, wherein environment mappings provide platform independence of test case and the abstract representation provides a platform-independent representation of test case, and wherein test scripts are generated for multiple test execution environments~~.

44. (currently amended) The system of claim 43, ~~further comprising:~~

wherein the first set of instructions comprises a syntax analyzer ~~for incoming scripts~~.

45. (currently amended) The system of claim 44, wherein the syntax analyzer is configured to generates a parse tree in the form of an Abstract Syntax Tree ~~(AST)~~ based on the at least one test case.

46. (currently amended) The system of claim 45, ~~further comprising:~~

~~logic~~ wherein the first set of instructions, when executed by the processor, further configures the processor to ~~implement semantic analysis and~~ convert the Abstract Syntax Tree ~~AST~~ to ~~an~~ the abstract ~~test case~~ representation based on an Application Object Model ~~(AOM)~~.

47. (currently amended) The system of claim 43, ~~further comprising:~~

~~logic~~ wherein the first set of instructions, when executed by the processor, further configures the processor to ~~for~~ enrich~~ing~~ the abstract ~~test case~~ representation ~~with~~ using information from an application metadata repository.

48. (currently amended) The system of claim 43, ~~further comprising:~~

~~logic for separating~~ wherein the first set of instructions, when executed by the processor, further configures the processor to separate the ~~application object~~ attributes and the input data from the at least one external interaction ~~sequencing~~ sequence to provide automatic parameterization.

49. (cancelled).